1. **Write a program to print fibonacci series using recursion.**

**Program:**

def fibonacci(n):

if n<=1:

return n

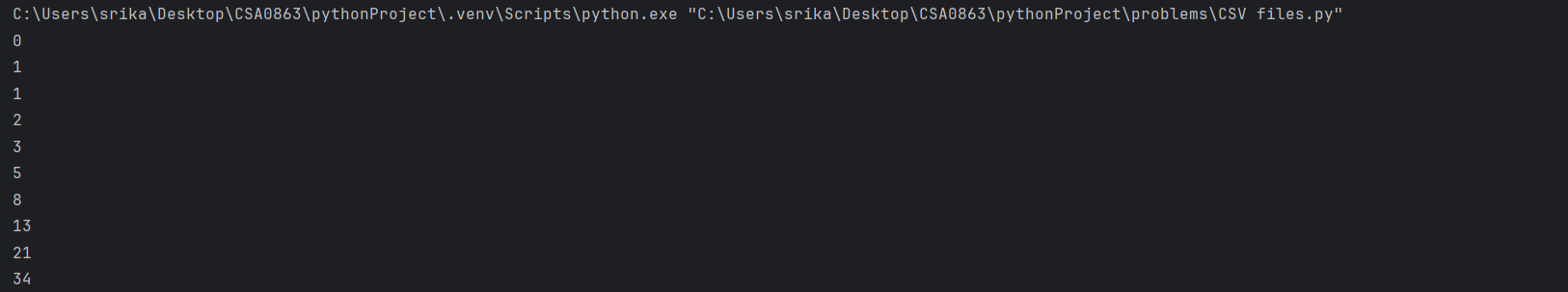
else:

return fibonacci(n-1)+fibonacci(n-2)

for i in range(10):

print(fibonacci(i))

**Output:**

Time complexity:

2.**Write a program to check the given no is Armstrong or not using recursive function.**

Program:

def count\_digits(n):

if n==0:

return 0

return 1+count\_digits(n//10)

def is\_armstrong\_recursive(n,num\_digits):

if n==0:

return 0

return(n%10)\*\*num\_digits+is\_armstrong\_recursive(n//10,num\_digits)

def is\_armstrong(n):

num\_digits=count\_digits(n)

return n==is\_armstrong\_recursive(n,num\_digits)

num=141

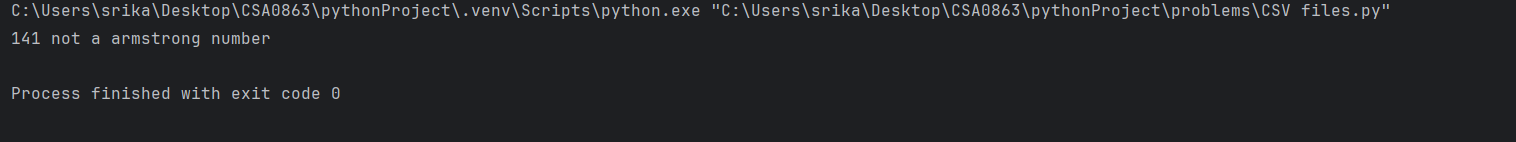
if is\_armstrong(num):

print(num,"is a armstrong number")

else:

print(num,"not a armstrong number")

Ouput:



3.**Write a program to find the GCD of two numbers using recursive factorization.**

Program:

def gcd\_recursive(a,b):

if b==0:

return a

else:

return gcd\_recursive(b,a%b)

num1=10

num2=20

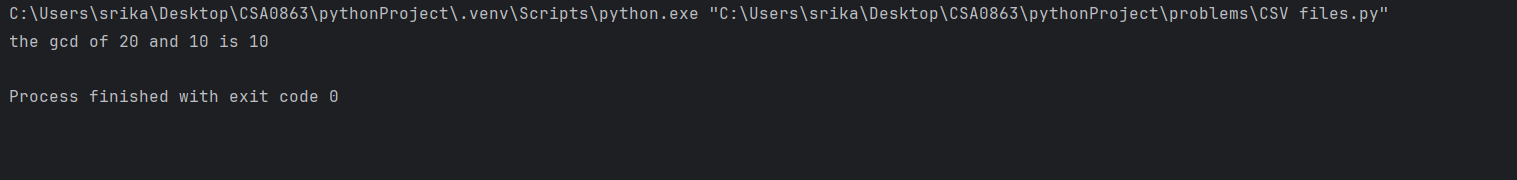
if num1<num2:

num1,num2=num2,num1

gcd=gcd\_recursive(num1,num2)

print("the gcd of",num1,"and",num2,"is",gcd)

Output:



Time complexity:

1. **write the program to find the largest number of an array.**

Program:

def find\_largest(arr):

if len(arr)==0:

return None

elif len(arr)==1:

return arr[0]

else:

max\_elements=arr[0]

for elements in arr[1:]:

if elements>max\_elements:

max\_elements=elements

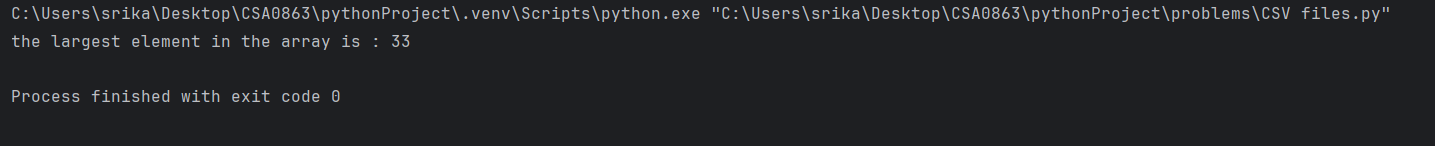
return max\_elements

array=[10,33,5,17,89]

largest=find\_largest(array)

print("the largest element in the array is :",largest)

Output:



Time complexity:

1. **Write a program to find the Factorial of a number using recursion.**

Program:

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n - 1)

def main():

num = int(input("Enter a number to find its factorial: "))

if num < 0:

print("Factorial is not defined for negative numbers.")

else:

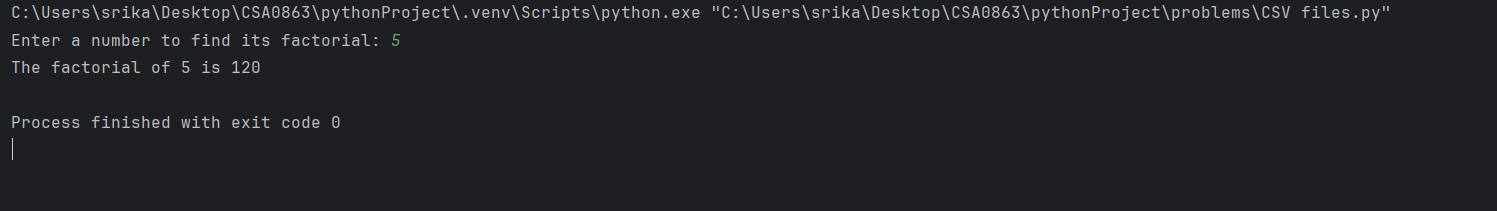
fact = factorial(num)

print("The factorial of", num, "is", fact)

if \_\_name\_\_ == "\_\_main\_\_":

main()

Output:



Time complexity:

6.**Write a program for to copy one string to another using recursion**

Program:

def copy\_string(source, destination, index=0):

if index == len(source):

return destination

else:

destination += source[index]

return copy\_string(source, destination, index + 1)

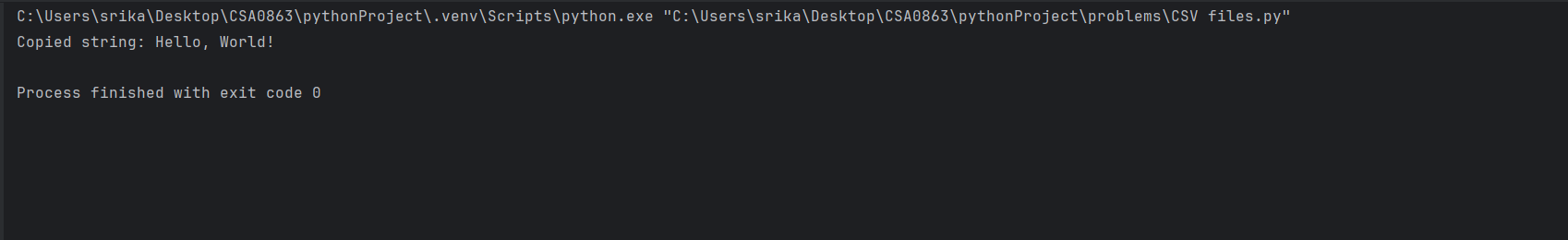
source\_str = "Hello, World!"

destination\_str = ""

result = copy\_string(source\_str, destination\_str)

print("Copied string:",result)

Output:



Time complexity:

7.**Write a program to print the reverse of a string using recursion**

Program:

def reverse\_string(source):

if len(source) == 0:

return ""

else:

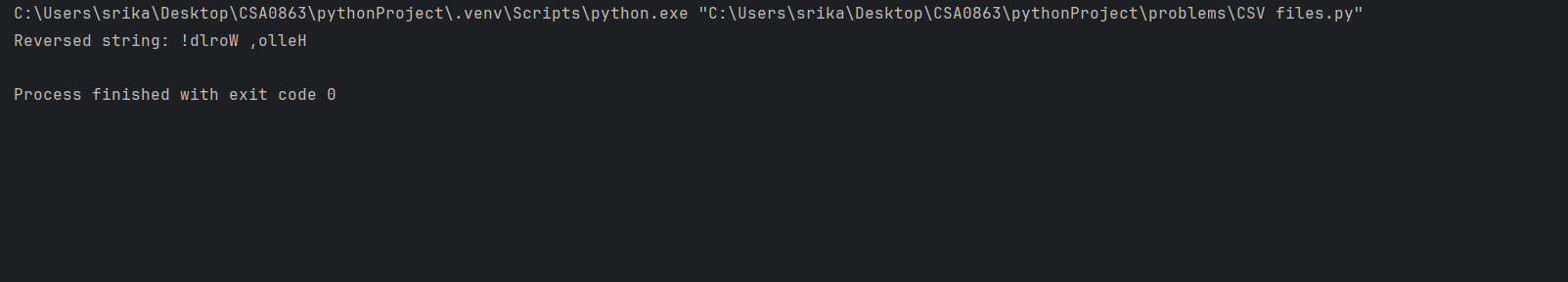
return source[-1] + reverse\_string(source[:-1])

source\_str = "Hello, World!"

result = reverse\_string(source\_str)

print("Reversed string:",result)

Output:



Time complexity:

8. **Write a program to generate all the prime numbers using recursion**

Program:

def is\_prime(n, divisor=2):

if n <= 2:

return n == 2

if n % divisor == 0:

return False

if divisor \* divisor > n:

return True

return is\_prime(n, divisor + 1)

def generate\_primes(start, end):

if start > end:

return

if is\_prime(start):

print(start)

generate\_primes(start + 1, end)

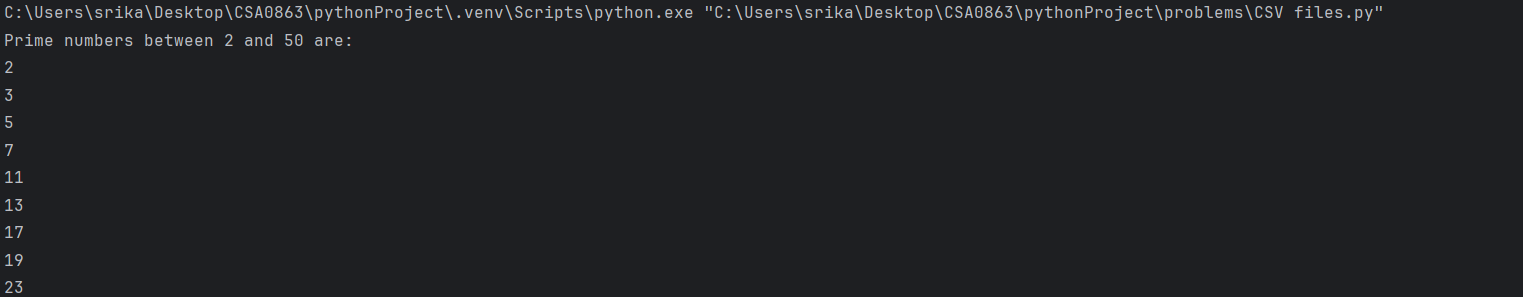
start\_num = 2

end\_num = 50

print("Prime numbers between", start\_num, "and", end\_num, "are:")

generate\_primes(start\_num,end\_num)

Output:



Time complexity:

9.**Write a program to check a number is a prime number or not using recursion.**

Program:

def is\_prime(n, divisor=2):

if n <= 1:

return False

if n == 2:

return True

if n % divisor == 0:

return False

if divisor \* divisor > n:

return True

return is\_prime(n, divisor + 1)

num = 17

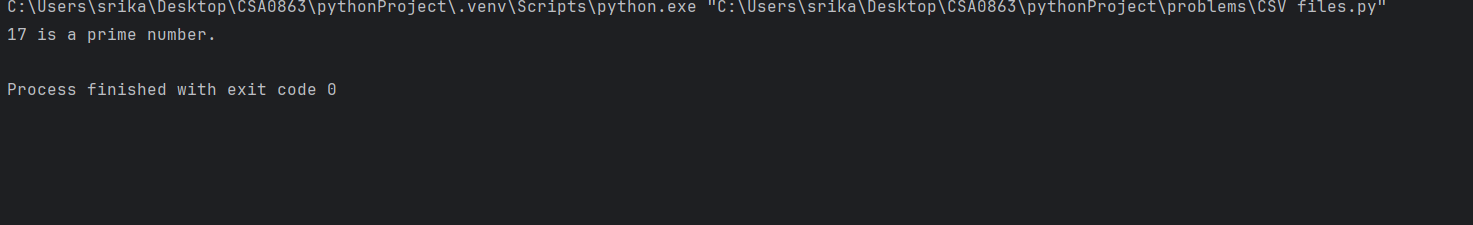
if is\_prime(num):

print(num, "is a prime number.")

else:

print(num, "is not aprimenumber.")

Output:



Time complexity:

1. **Write a program for to check whether a given String is Palindrome or not using recursion**

Program:

def is\_palindrome(s):

s = s.lower() # Convert string to lowercase for case-insensitive comparison

if len(s) <= 1:

return True

if s[0] != s[-1]:

return False

return is\_palindrome(s[1:-1])

input\_str = "Madam"

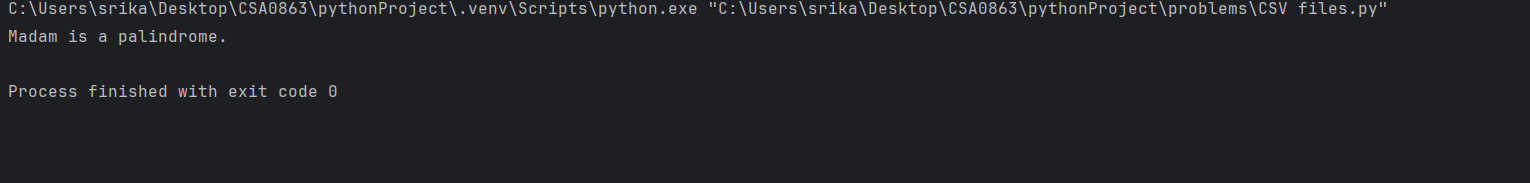
if is\_palindrome(input\_str):

print(input\_str, "is a palindrome.")

else:

print(input\_str, "is notapalindrome.")

Output:



Time complexity: